

A General-Purpose Display Processing and Tutorial System

K. J. ENGVOLD AND J. L. HUGHES

IBM Education Center, Poughkeepsie, N. Y.

ADEPT (A Display-Expedited Processing and Tutorial) system is described. This system was designed to improve man-computer communications by employing a display unit to interleave tutoring with other computer operations such as simulation, programming, and information retrieval. It is written in FORTRAN IV (G) for the IBM System/360, Model 40, and the IBM 2250 Display Unit under Operating System/360. ADEPT is a cataloged program that controls the standard operating system by terminating and rescheduling itself automatically, relinquishing computer resources allocated to it, and surrendering control to the operating system to perform other jobs. It expands the power and flexibility of computer-assisted instruction by making immediately available to students, teachers, and other users, the full resources (system-cataloged programs) of the operating system.

Language processors and compilers, simulation models, mathematical solution techniques, stored data, and all other library and user programs can be incorporated into instructional material without reprogramming. Illustrations of the various applications are presented and their implications are discussed.

KEY WORDS AND PHRASES: computer-assisted instruction, tutorial systems, programming, simulation, modeling, information retrieval, operating systems, graphics, displays, man-machine interface, on-line computing, graphic programming

CR CATEGORIES: 1.5, 3.3, 3.5, 3.8, 4.0, 4.3

Introduction

The typical computer teaching system has been designed as special purpose, stand-alone software. Its principal function is to interact with the student during the presentation of drill or tutorial material at a typewriter or display terminal. It also provides a means by which authors unfamiliar with computer programming can prepare drill or instructional sequences for presentation to students. Descriptions of representative teaching systems developed for CDC, DEC, and IBM computers are given in [4], and similar systems have been developed for GE, RCA, and other computers [5].

These special purpose systems are limited to performing functions which have been specially programmed within the teaching system. Adding other features to make the teaching system more flexible, i.e. capable of doing more than just presenting course material, therefore requires extensive additional programming. For example, if one wants the student to write and execute computer programs at intervals during a programming course, a special language processor or compiler must be programmed within the teaching system. If simulation is needed to illustrate topics covered in the instructional sequence, the model must also be programmed specifically to operate within the teaching system.

It is therefore suggested that the task of designing a tutorial system be viewed from a broader perspective. Much of the software potentially useful in the teaching process, such as language processors, compilers, models, and other user programs, has already been programmed for the standard operating systems of large computers. The scope and effectiveness of the teaching process would be greatly increased if all this software could be interleaved with the teaching system without costly reprogramming. Conversely, the teaching system would prove invaluable in teaching or guiding the user to understand and operate the extensive program library of the operating system. Thus both activities, data processing and teaching, would profit if they could be combined under a single system.

Based on these considerations, a new, general purpose model of a computer processing and teaching system is proposed. In order to improve man-computer communication, the computer would be controlled from a display unit. All the resources of the operating system, including extensive tutorial and reference material, would be called upon and executed by pointing a lightpen at the display. This would permit these materials to be freely interleaved with calls to any of the cataloged programs of the standard operating system, such as processors or models. Under author control, the student or user would have the full power of the operating system at his command in learning any subject or in operating the computer.

To accomplish this result, the display unit would be programmed as the interface between the user and the operating system. The program would be cataloged on disk under a standard operating system and would permit text and graphic material—for instructing or guiding the user, for programming, or for information retrieval, etc.—to be prepared easily for the display. This cataloged program would automatically terminate and reschedule itself,

turn control over to the operating system in order to process any other desired program(s), and ultimately regain control in order to proceed with the instruction or guidance material.

In effect, the operating system would function as a set of subroutines under control of the display program once the program is in execution. Thus all features added to the standard operating system in the future would automatically come under control of this program without the need for extensive additional programming.

The language and software designed to accomplish these objectives were developed for the IBM 7044 System and are described in [1]. This report describes the characteristics of an expanded OS/360 experimental version and illustrates its applicability to tutoring, simulation, programming, and information retrieval.

Description of System

The ADEPT (A Display-Expedited Processing and Tutorial) system was developed experimentally for the IBM System/360, Model 40, and the 2250 Display Unit, Model 1. It presently operates under Operating System/360 (Option 2, Version 9.5) and requires all four partitions (180K bytes) of storage as well as two 2311 disk storage devices. The 2250 has an 8K buffer, and a lightpen, alphanumeric keyboard and program function keyboard.

The ADEPT system consists of:

1. A language containing control and text codes which operate in one of three modes (author, user, and programmer) that are freely interchangeable.
2. An interpreter program that processes programmer, author, and user control codes and automatically switches the system from one mode to another.



FIG. 1. Page of text area showing author text and control codes

3. Routines that automatically:

- a. Checkpoint ADEPT, i.e. terminate and reschedule it for a restart later;
- b. Relinquish computer resources allocated to ADEPT;
- c. Transfer control to the operating system for processing and execution of a new job (for example, assembly, compilation, and simulation); and
- d. Restart the ADEPT job when (b) is completed.

ADEPT has 37 FORTRAN IV (Level G) and 2 assembly language subroutines. The FORTRAN programs call upon 40 library and GPAK (Graphic Package, Version II) routines [6].

Characteristics of the Display. Text and drawings presented at the display are stored as a series of page images (45 lines, 64 characters) on disk. One page at a time is read from disk into the display buffer for presentation at the terminal. At present, 100 pages are provided for the text area on disk, as well as 100 pages for graphic material. In addition, 100 pages of programmer area are allocated on disk. The number of pages available can easily be increased if required.

ADEPT Applications

Tutoring. By signing on in Author Mode, the author can enter course or reference material by punch card or by typing at the display (Figure 1). He uses a freely movable cursor to position characters anywhere on a page. To identify the different categories of text material, the author uses text codes, such as QU, CA, WA, and BR, similar to those of the IBM 1440-1050 Coursewriter [3]. In addition, text codes to permit lightpenning and interleaving the text with any system-cataloged program were created. Thus the text code LP (lightpen) identifies words (TECH, MODEL, RETURN) to be displayed for lightpenning by the user in order to change the display. BP (Branch-to-Program) branches the user to a page which contains system control statements to execute any system cataloged program after terminating ADEPT. When the user has finished working with this program, he lightpens RETURN to terminate it and to resume the instructional sequence under ADEPT.

The author controls the display by lightpenning author control codes, shown at the right of the display (Figure 1). Pointing to these control codes instead of typing them enables him to work faster with less chance of error. For example, after the author has completed typing an entire page of text, he lightpens RECORD to store it on disk. By lightpenning other control codes, the author can ADVANCE or BACKUP one page at a time, GO TO any specified page, INSERT, DELETE, PRINT any page(s), and SIGN OFF. When needed to indicate a page number, e.g. after GO TO is lightpenned, a subsidiary panel containing a number keyboard appears. A DRAW control code permits the author to draw static or animated line drawings with the lightpen to accompany the text. By lightpenning TEST, he can enter USER Mode to try out the material he has prepared (Figure 2).

The author can thus prepare tutorial and reference material interspersed with calls upon the operating system to execute simulation, data processing, and information retrieval functions. ADEPT therefore provides great flexibility and power to the author in designing course material. In mathematics and science, for example, access to a computer for simulation and programming during an instructional sequence has obvious advantages. References to academic subjects are mentioned only in passing because this report is concerned primarily with applications in the computing field. As these are presented, however, the advantages of ADEPT for more traditional academic areas should become clear.



FIG. 2. Text area as seen by user

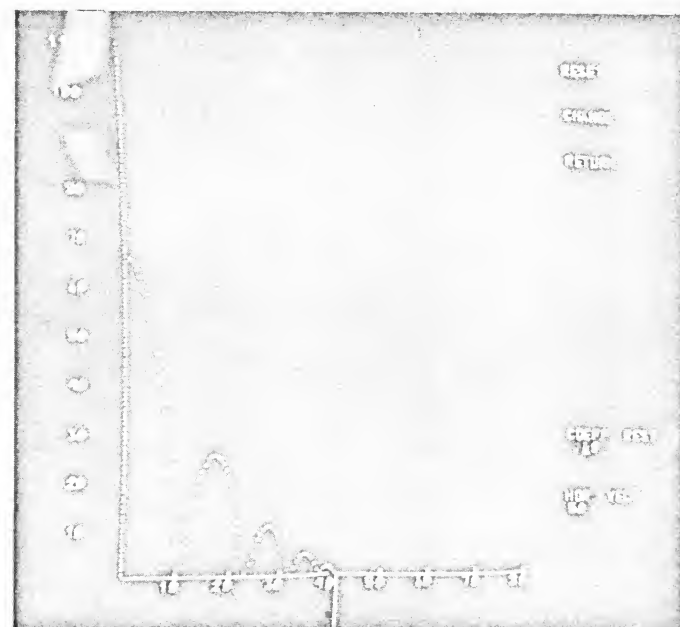


FIG. 4. Display of solution to original Bouncing Ball problem



FIG. 3. Initial display of Bouncing Ball model

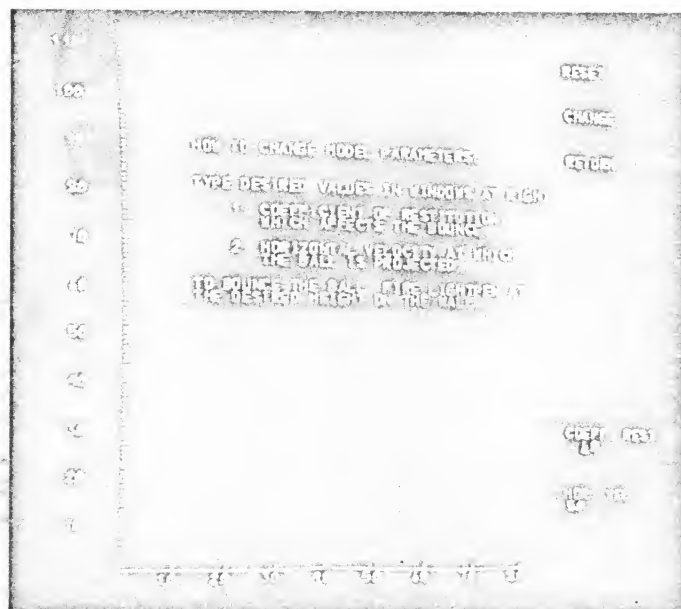


FIG. 5. Display of instructions for changing Bouncing Ball model parameters

and the computer is lost. By means of the display and its tutorial capability, ADEPT overcomes this problem, particularly for persons unfamiliar with computers. A model of the Bouncing Ball problem, frequently used in executive computer concepts courses, will serve as an illustration.

In User Mode the executive is first presented with a general explanation of the problem in the text area (Figure 2). If he desires a more technical explanation, he lightpens TECH to proceed to another page of the display. Otherwise, he lightpens MODEL to view a display of the model (Figure 3) of the cataloged Bouncing Ball program. By following the directions, he repeatedly uses the lightpen to operate the model, i.e. bounce the ball, until he achieves a solution to the original problem (Figure 4).

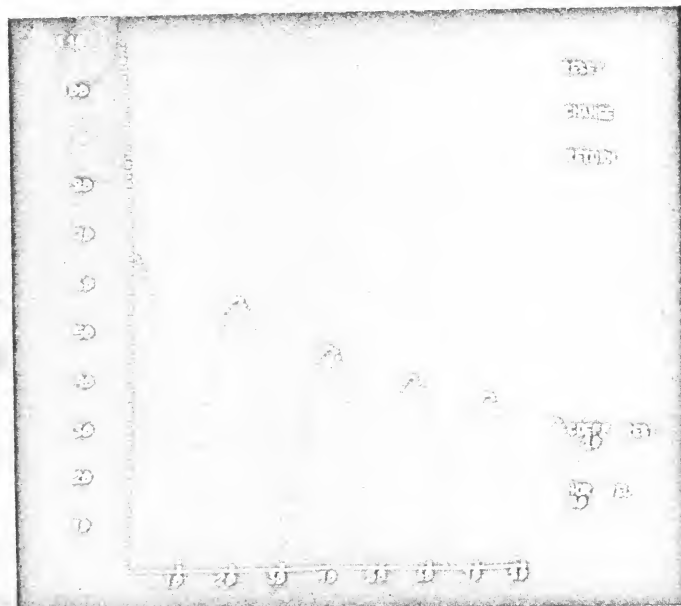


FIG. 6. Solution to Bouncing Ball problem after increasing coefficient of restitution parameter

TABLE OF CONTENTS FOR OS/360 UTILITY PROGRAMS	
1. OS/360 UTILITY PROGRAMS	1
2. OS/360 UTILITY PROGRAMS	2
3. OS/360 UTILITY PROGRAMS	3
4. OS/360 UTILITY PROGRAMS	4
5. OS/360 UTILITY PROGRAMS	5
6. OS/360 UTILITY PROGRAMS	6
7. OS/360 UTILITY PROGRAMS	7
8. OS/360 UTILITY PROGRAMS	8
9. OS/360 UTILITY PROGRAMS	9
10. OS/360 UTILITY PROGRAMS	10

FIG. 7. Table of Contents for OS/360 utility programs

TABLE OF CONTENTS FOR FORTRAN PROGRAMS	
1. FORTRAN PROGRAMS	1
2. FORTRAN PROGRAMS	2
3. FORTRAN PROGRAMS	3
4. FORTRAN PROGRAMS	4
5. FORTRAN PROGRAMS	5
6. FORTRAN PROGRAMS	6
7. FORTRAN PROGRAMS	7
8. FORTRAN PROGRAMS	8
9. FORTRAN PROGRAMS	9
10. FORTRAN PROGRAMS	10

FIG. 8. Table of Contents for FORTRAN programs

INSTRUCTIONS FOR PROCESSING FORTRAN PROGRAMS	
1. INSTRUCTIONS FOR PROCESSING FORTRAN PROGRAMS	1
2. INSTRUCTIONS FOR PROCESSING FORTRAN PROGRAMS	2
3. INSTRUCTIONS FOR PROCESSING FORTRAN PROGRAMS	3
4. INSTRUCTIONS FOR PROCESSING FORTRAN PROGRAMS	4
5. INSTRUCTIONS FOR PROCESSING FORTRAN PROGRAMS	5
6. INSTRUCTIONS FOR PROCESSING FORTRAN PROGRAMS	6
7. INSTRUCTIONS FOR PROCESSING FORTRAN PROGRAMS	7
8. INSTRUCTIONS FOR PROCESSING FORTRAN PROGRAMS	8
9. INSTRUCTIONS FOR PROCESSING FORTRAN PROGRAMS	9
10. INSTRUCTIONS FOR PROCESSING FORTRAN PROGRAMS	10

FIG. 9. Instructions for processing FORTRAN programs

It should be noted that the Bouncing Ball model was already programmed and cataloged to run under OS/360. By means of the BP (Branch-to-Program) text code (Figure 1), this program was linked to text explaining the model. In a similar fashion, any other economic, mathematical, or scientific model programmed under OS/360 could be explained and operated at the display unit. No assistance from professional computer people is necessary. ADEPT thus gives users unsophisticated in computing the ability to interact with the computer immediately and knowledgeably in simulation applications, thereby increasing their efficiency and productivity in this kind of problem-solving.

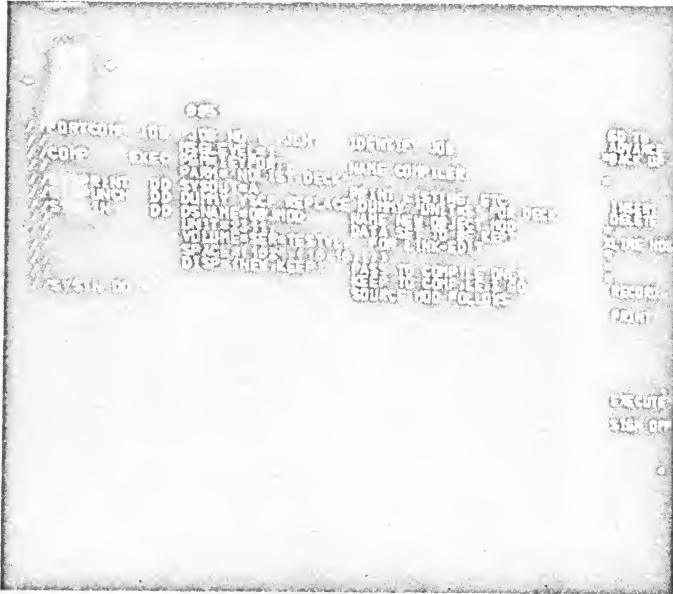


Fig. 10. OS/360 Job Control Language cards for FORTRAN compilation

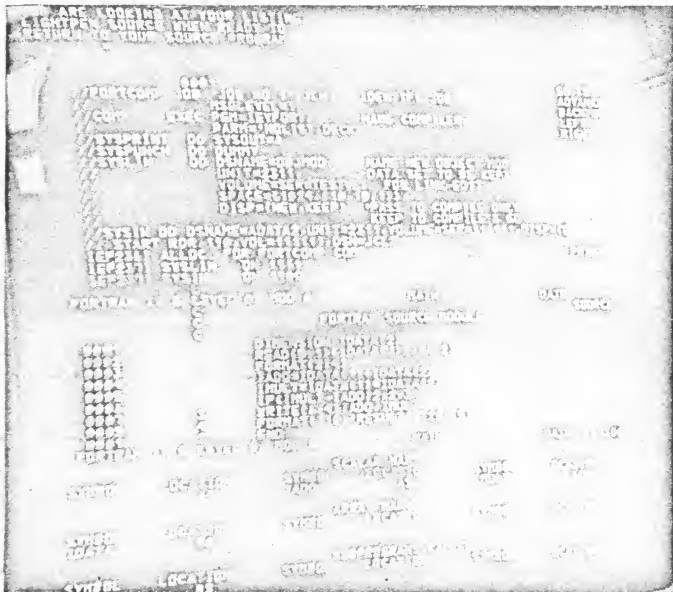


Fig. 11. First page of FORTRAN machine listing on the display

Programming. The operating system of a large computer contains a library of programs to carry out many different functions. If the user could quickly call upon these functions from a display unit without recourse to keypunching and inserting cards in the job stream, man-computer communication would be greatly improved.

Programmer Mode makes it possible to accomplish this. The programmer types (or reads from cards) into the programmer area the appropriate OS/360 Job Control Language and data card decks to accomplish different operations. He also sets up a table of contents, listing the operations and associated page numbers where the decks are stored (Figure 7).

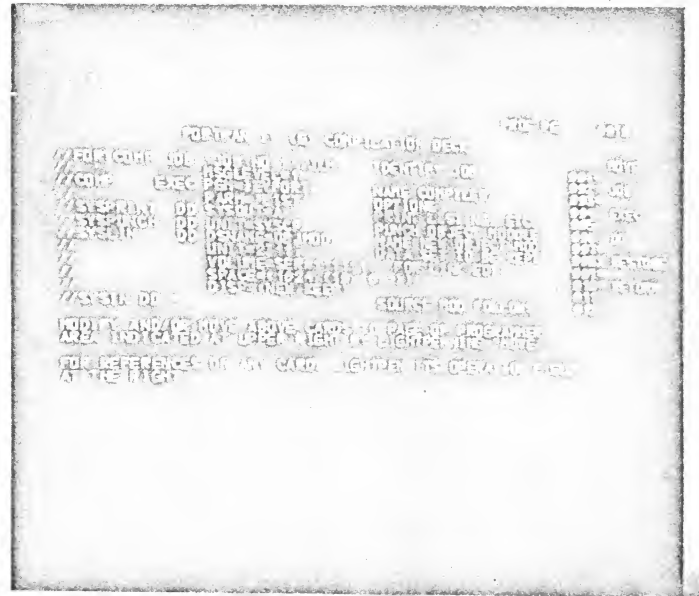


Fig. 12. OS/360 Job Control Language deck stored in user reference area



Fig. 13. References explaining DD Job Control Language card

Suppose the programmer, after scratching a data set, wants to list the volume table of contents of a disk pack (LIST VTOC in Figure 7). He would execute the program as follows:

1. Lightpen EXECUTE and a new panel containing numbers would appear.
 2. Lightpen 63, the page number containing the LIST VTOC program.
 3. Lightpen EXECUTE again to begin processing and to obtain a VTOC listing on the display or printer.
- Thus, merely by pointing his lightpen at the screen, the programmer can immediately execute any previously programmed application. He can also easily change parameters in any programs by going to the appropriate page and using the cursor and keyboard.

Another example: Suppose the programmer wants to compile, link-edit, and execute a source program in any language processed by the operating system. In the case of FORTRAN IV (G), the table of contents (Figure 8) indicates that these operations are covered on pages 50-59. Page 50 contains instructions (Figure 9) for, and page 51 (Figure 10) and pages 58 and 59 contain the OS/360 Job Control Language cards for compilation, linkage-editing, and execution, respectively. Pages 52-57 are blank pages reserved for FORTRAN source code. The programmer would:

1. Lightpen GO TO page 52 and begin typing his FORTRAN source program in standard coding format, or read in his card deck by means of an ADEPT utility program.
2. He could also GO TO a page containing the JCL cards in order to change a parameter, or INSERT or DELETE a line.
3. Finally, he would process his program by lightpenning EXECUTE, the page numbers 51 and 59, and EXECUTE again. This would produce a machine listing on the display (Figure 11).

After scanning this listing and debugging his source program at the display, he can repeat the above procedure until the program runs correctly. At any time he can also obtain a printer listing instead of the display listing. The entire programming sequence for standard features of the operating system can thus be accomplished without handling, punching, or reading in cards.

Information Retrieval. Modern systems programming requires a memory for the countless details of machine operation and system control card parameters, formats, and sequencing, all of which strain the recall of even the most experienced programmers. If extensive reference material on these topics could be quickly displayed to the user, he could use his machine time—always a scarce commodity—more efficiently than at present.

One technique for accomplishing this under ADEPT is to use Author Mode to store a library of standard sets of system control cards in the text area. These card decks could be supported by reference materials explaining the functions of the various card parameters. Any deck could

be displayed by lightpenning its number in a table of contents.

For example, Figure 12 presents a basic deck of OS/360 JCL cards needed for a FORTRAN IV (G) compilation. If the programmer has forgotten some detail about a card parameter, he can lightpen the operation field of that card (JOB, EXEC, DD) and display pertinent reference material (Figure 13).

After reading the references on a card, he can type changes in the card at the top of the reference page. Lightpenning RECORD causes the change to be made in the standard deck, which is then displayed in modified form. Finally, when the programmer has modified the deck to his satisfaction, he lightpens MOVE to store it in the programmer area for execution.

In a similar fashion, explanations of all features of the operating system programming languages and of machine operation could be stored in the text area for instant retrieval by lightpen. ADEPT can therefore provide instant technical support to the programmer and machine operator as well as training materials for the machine-room trainee.

Summary

By means of the lightpen, the experimental display system described makes it easy to interleave tutorial with other software already programmed for a standard operating system. Thus a wide range of functions can be provided at the display terminal—even to those unfamiliar with computing—by drawing upon the massive existing software resources of the operating system. Examples from such areas as tutoring, simulation, programming, and information retrieval have been given in order to illustrate the range of applications. In the scope and flexibility of its terminal activity, ADEPT resembles systems like Project MAC at MIT [2] rather than most special purpose teaching systems [4], although Project MAC was primarily designed for problem-solving rather than teaching. ADEPT, by making the tutorial feature an integral part of computing, suggests a means of improving man-machine communication that merits further exploration.

RECEIVED MARCH, 1968; REVISED MAY, 1968

REFERENCES

1. ENGVOLD, K. J., AND HUGHES, J. L. A model for a multifunctional teaching system. *Comm. ACM* 10, June (1967), 339-342.
2. FANO, R. M. The MAC system: a progress report. In *Computer Augmentation of Human Reasoning*, Sass and Wilkinson (Eds.), Spartan Books, New York, 1965.
3. SCHWARTZ, H. A., AND LONG, H. A. Instruction by computer. *Datamation* 12 (Sept. 1966), 73-90.
4. ZINN, K. L. Computer assistance for instruction: a review of systems and projects. In *The Computer in American Education*, Bushnell and Allen (Eds.), Wiley, New York, 1967.
5. *Automated Education Letter* 2 (July-Aug. 1967), 5-15.
6. GPAK: An on-line System/360 graphic data processing package with real-time 2250 input and display. S/360 General Program Library, 360D-3.4.005, Sept. 26, 1966.